Sousanna Chugunova
CMSC204
Dr.Thai

Project 2 Write Up

Approach

Because this project had many parts and requirements, I started by planning out my approach before writing any code. I organized my thoughts in a document to make sure I didn't miss anything important.

Once I had everything mapped out, I created the necessary classes for the project.

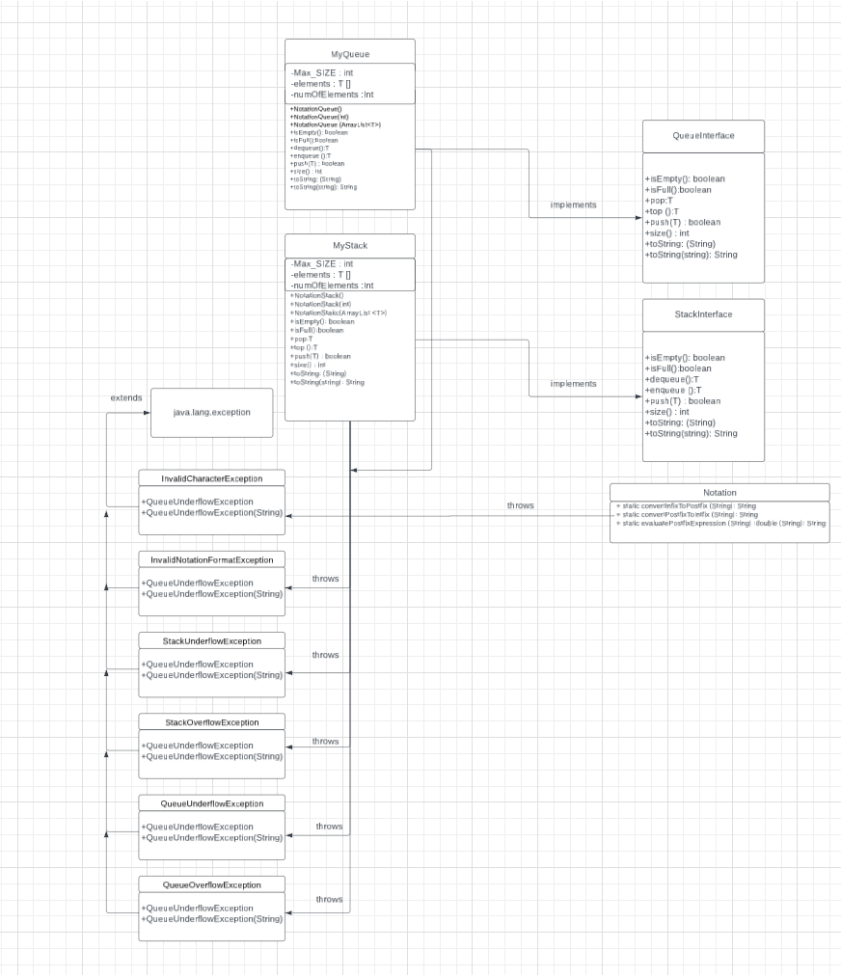
After setting up all the classes, I made a diagram to visualize how each part of the project fit together. This helped me understand how everything should work as a whole.

With the structure in place, I focused on coding the Notation class and testing it thoroughly. During testing, I ran into issues with extra spaces in expressions, which affected the output. I also had to ensure that the results were integers, not decimals.

After several rounds of testing and adjustments, I managed to pass all the tests successfully. Seeing the GUI display the correct results was a satisfying moment after putting in a lot of effort.

Overall, this project taught me the importance of planning ahead, being meticulous in coding, and testing thoroughly to catch and fix any issues along the way.

Design - UML Diagram (helped me keep organized)



Algorithm

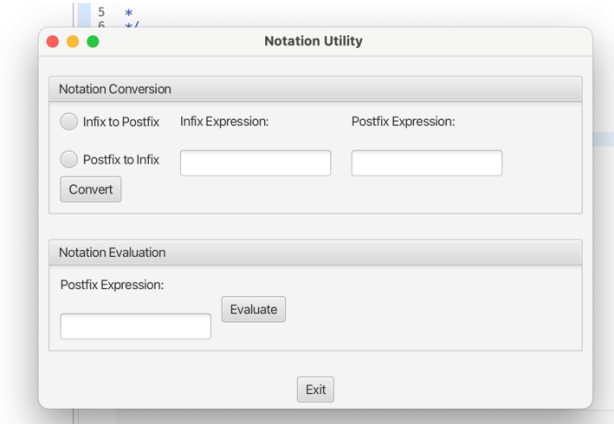
To implement the Notation utility class, I'm focusing on three main tasks: converting infix to postfix expressions, converting postfix back to infix, and evaluating postfix expressions.

When converting infix to postfix, I'm carefully going through the expression, dealing with numbers directly and using a stack to handle operators based on their importance. I also manage parentheses to group operations correctly. For converting postfix back to infix, I analyze the postfix expression using a stack to build the infix notation, ensuring each operation is placed inside parentheses as needed. When evaluating postfix expressions, I went through the expression and using a stack to perform calculations step by step to get the right results.

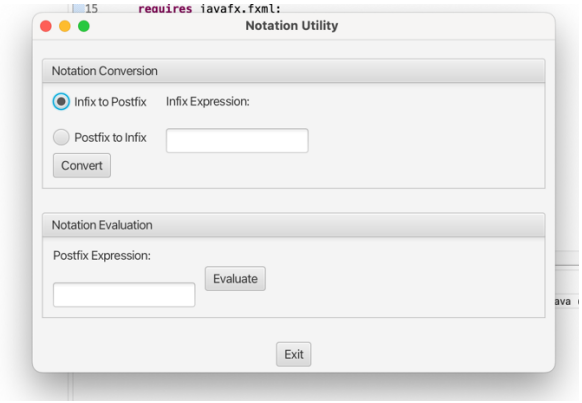
To handle any issues like incorrect characters or mismatched parentheses, I've included exception handling. To ensure everything works well, I'm testing extensively with JUnit using different inputs. Additionally, I'm using a GUI to interactively validate the functionality.

Test plans

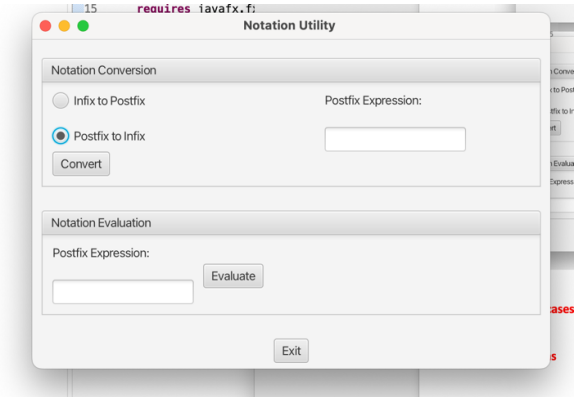
At startup:



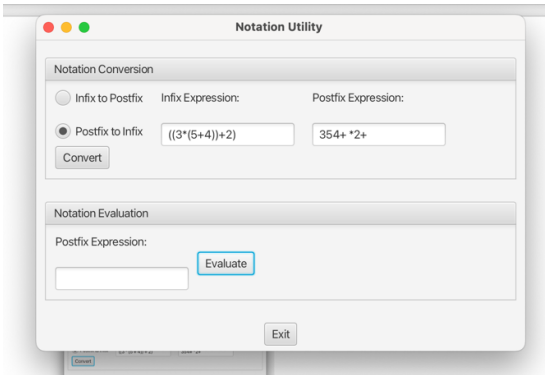
After selecting infix to postfix:



After selecting postfix to infix:

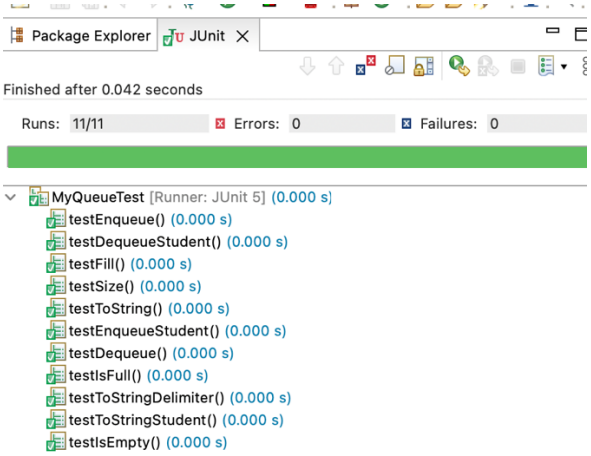


Evaluate:

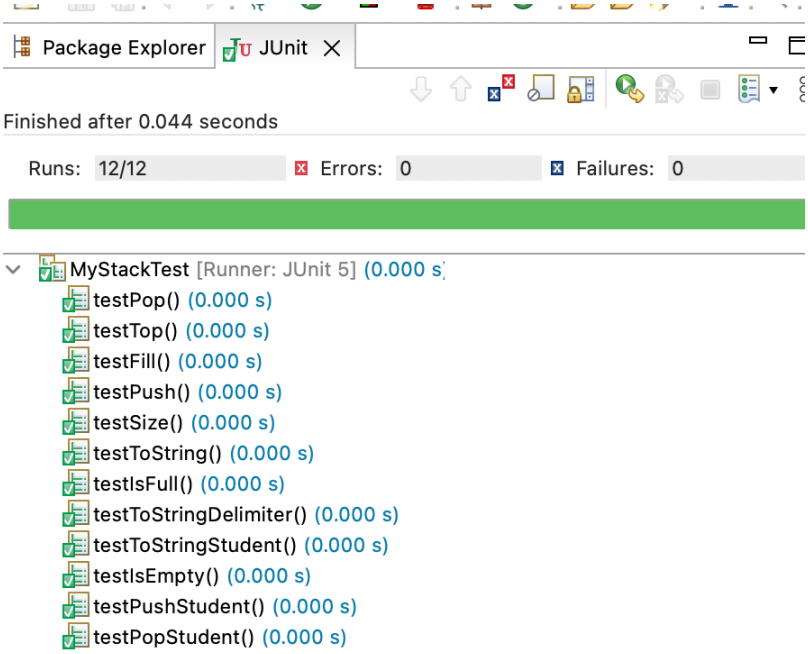


Junit test cases

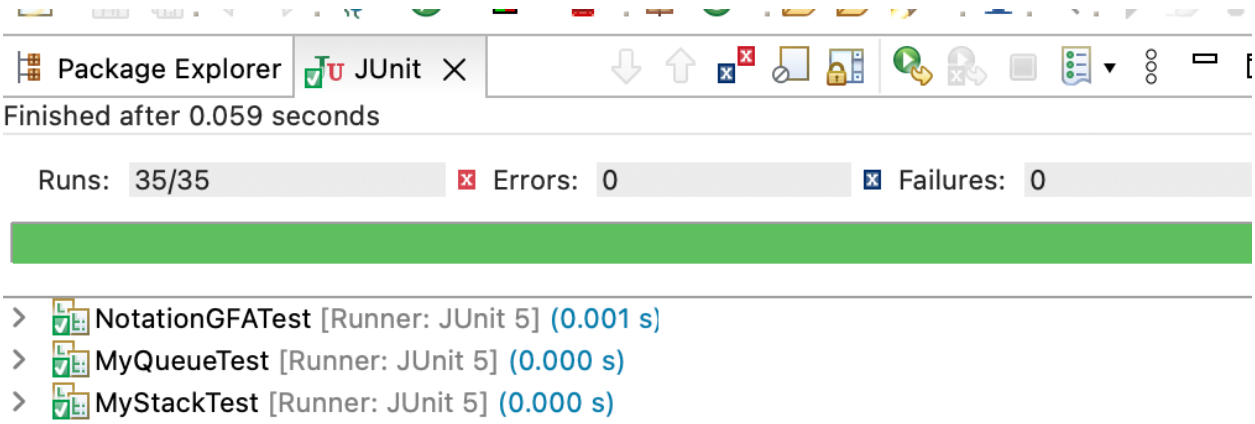
MyQueueTest



MyStackTest



NotationGFATest



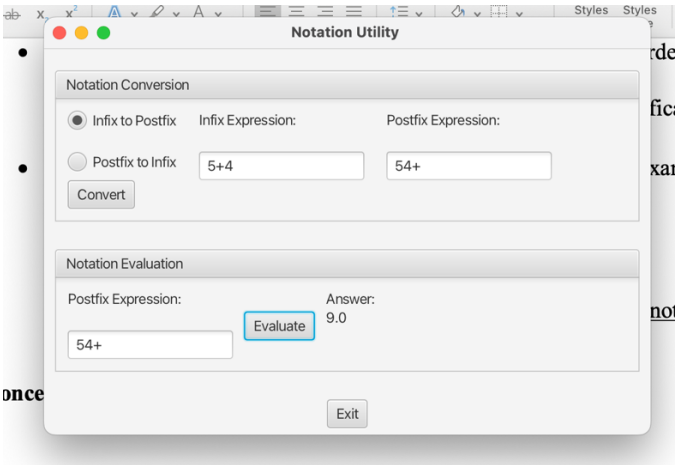
JavaFX runs

What were the actual outputs (in the JavaFX GUI) when you ran it?
What were the expected vs. actual outputs?

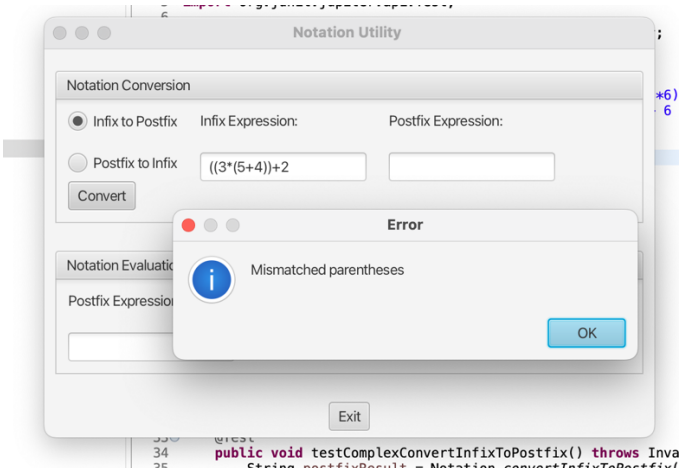
(See below)

Addition:

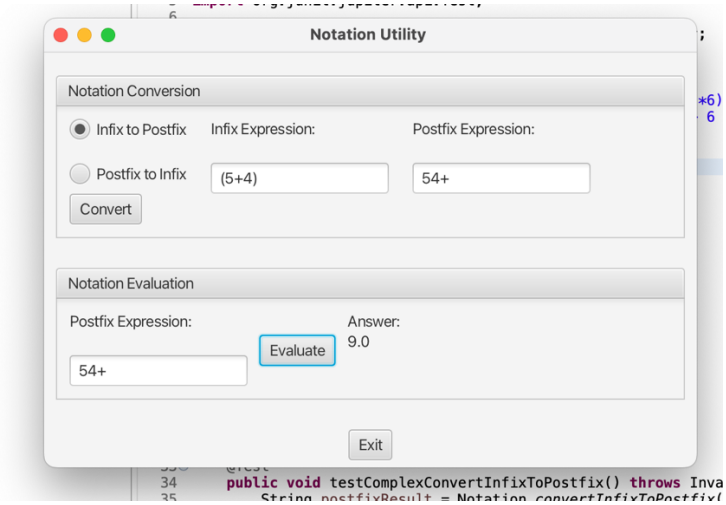
Expected: 9.0
Actual: 9.0



Mismatched Parenthesis:
Expected: Error msg (Mismatched Parenthesis)
Actual: Mismatched Parenthesis



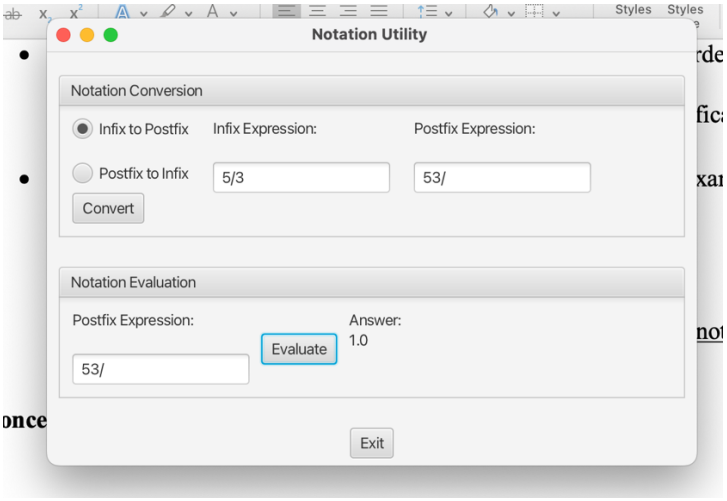
Matching parenthesis:
Expected: 54+
Actual: 54+



Division:

Expected: 53/

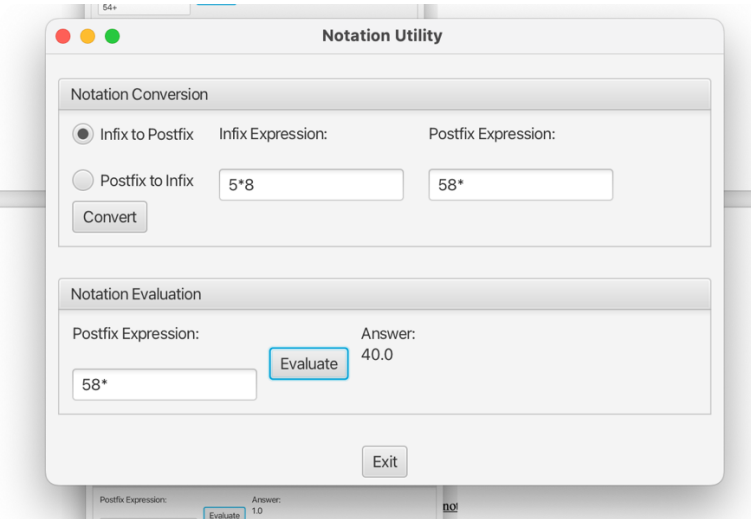
Actual: 53/



Multiplication:

Expected: 58*

*Actual: 58**



What test cases did you run?
-MyStackTest
-MyQueueTest

Will your project be able to pass a set of private test cases? Yes

Highlight your learning experience and lessons learned

- This project has been quite a journey, involving a lot of work and attention to detail. I've learned that building utilities like converting expressions and evaluating them requires careful planning and testing. It's not just about writing code but also about making sure everything works smoothly, which can (and was) challenging. The process has taught me a lot about how to handle different parts of a program and the importance of thorough testing to ensure everything functions as expected. Overall, it's been a valuable experience in understanding what it takes to create reliable software tools.

Assumptions that you made

- I assumed this project would take me a long time and it did. I knew there were concepts I was not familiar with and I knew it would take me some time to research, study and finally start coding. I went into this project with this mindset which made it easier to digest and not get as overwhelmed when methods, tests, and classes were giving me trouble.

Anything else that I need to know?

-I had to change the NotationGFATest name back to NotationGFATest from NotationTest because when I imported it was NotationTest
-Had to define the size in the StackInterface
-Screen pops up when attempting to run GUI, must use NotationGUI-notation
-When attempting to use %, my computer keeps spinning and nothing loads as it does with other operators. I think this might be a mac issue, please let me know if it is not
-I tried my best